An Online Handwriting Recognition System For Turkish

Esra Vural, Hakan Erdogan, Kemal Oflazer, Berrin Yanikoglu *
Sabanci University, Tuzla, Istanbul, Turkey 34956

ABSTRACT

Despite recent developments in Tablet PC technology, there has not been any applications for recognizing handwritings in Turkish. In this paper, we present an online handwritten text recognition system for Turkish, developed using the Tablet PC interface. However, even though the system is developed for Turkish, the addressed issues are common to online handwriting recognition systems in general.

Several dynamic features are extracted from the handwriting data for each recorded point and Hidden Markov Models (HMM) are used to train letter and word models. We experimented with using various features and HMM model topologies, and report on the effects of these experiments. We started with first and second derivatives of the x and y coordinates and relative change in the pen pressure as initial features. We found that using two more additional features, that is, number of neighboring points and relative heights of each point with respect to the base-line improve the recognition rate. In addition, extracting features within strokes and using a skipping state topology improve the system performance as well. The improved system performance is 94% in recognizing handwritten words from a 1000-word lexicon.

Keywords: online,handwriting, recognition, HMM, Turkish

1. INTRODUCTION

Online handwriting recognition and related applications received renewed interest with the introduction of the Tablet PC. Online handwriting is collected using a pressure sensitive tablet and a special pen and the captured pen position and pressure information is input to the online OCR system. In this work, we developed a prototype Turkish language module for the Tablet PC.

There are several research articles on online handwriting recognition[1–7] and the Tablet PC has online handwriting recognition support for more than 15 languages,[8] however we are not aware of such a system for Turkish. This is partly due to the difficulties associated with the agglutinative nature of the language. Typically, a fixed vocabulary is assumed in many handwriting recognition systems; for instance, a 30,000-word lexicon for English is sufficient for most applications. However, the agglutinative word structure in Turkish is an obstacle for having such a small sized lexicon, as explained in.[9] Yanikoglu and Kholmatov's system for unlimited vocabulary offline handwriting recognition system for Turkish uses a Turkish prefix recognizer developed by Oflazer and his group.[10] In this work, the first stage of an unlimited vocabulary system is developed using word list; however in the future this system will be similarly extended.

2. SYSTEM

The recognition system consists of training and test stages. In the training stage, the collected training data is used to train Hidden Markov Models (HMMs) representing handwriting units. HMM modeling is used extensively for both speech and handwriting recognition in the literature. We experimented with letter and word HMM models in this study.

2.1. User Interface

Input to the Tablet PC is via a pressure sensitive screen capturing the pen tip's x and y coordinates and pressure information. For every word, approximately 300 points are sampled. A user interface using the Tablet PC API has been developed to access the collected data by the Tablet PC.

*E-mail: esravural@su.sabanciuniv.edu, {haerdogan,oflazer,berrin}@sabanciuniv.edu

2.2. Hidden Markov Models

Handwriting data collected by a Tablet PC is trained with Hidden Markov models. HMMs are very successful in modeling speech and handwriting, where the visible observations and the hidden state transitions generating the observations form a doubly stochastic process with a sequential nature. HMMs are defined with a finite number of states where the features are assumed to be generated by a probability distribution depending on the current state or transition. The non-stationarity of the features is explained by the state transitions. The parameters of a Hidden Markov Model is as follows:

N: Number of States
$A = [a_{ij}]_{N\times N}$: State Transition Matrix
$b_j(o_t) = P(o_t, s_t = j)$: The observation probabilities at state j
o_t: The observation at time t
s_t: The state at time t
$\pi_i = P(s_1 = i)$ Initial probability at state i
λ: Whole set of model parameters.

For a particular model λ, the probability that the sequence of observations $O = [o_1 o_2 o_3 .... o_n]$ is generated by the state sequence $q = [s_1 s_2 s_3 .... s_n]$ is as follows:

$$P(O, q|\lambda) = P(O|q, \lambda)P(q|\lambda) = \prod_{i=1}^{T} b_{s_i}(o_i) \prod_{i=1}^{T-1} a_{s_i s_{i+1}}. \quad (1)$$

For all possible state sequences the probability of generating the observation sequence O in a particular model λ is shown below. While testing the system the most probable model is chosen as the most probable word.

$$P(O|\lambda) = \sum_{\forall q} P(O|q, \lambda)P(q|\lambda). \quad (2)$$

We used the HTK software for HMM training and recognition.[11] We experimented with word and letter based HMM models. When word models are used, one model is trained for each word in the lexicon, using several training samples of that word. When letter models are used, each word in the lexicon is labeled according to their constituent letter models and the letter models are trained using the well-known forward-backward algorithm. Then, a word model is created by consecutively lining up the letter models that represent the letters of that word.

The letter or word models in this work use a left-to-right topology. This topology is suitable to model the forward moving time dimension of the signal in speech and handwriting recognition. Hence, the state transition probabilities and the initial state probabilities are constrained as follows: $a_{ij} = 0, j < i$, $\pi_i = 0, i > 1$. We have trained these models with or without skipping topology. Figure 1 shows a left-to-right model with skipping state topology.

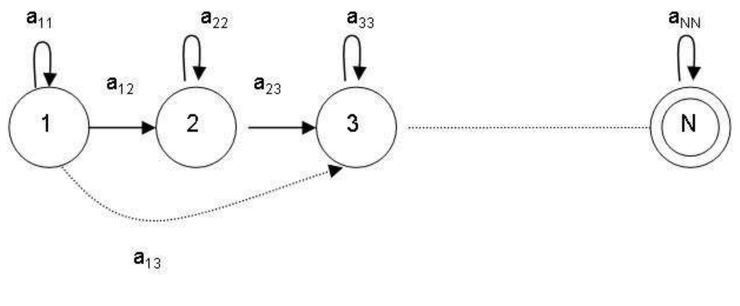

Figure 1. HMM representation for a left-to-right topology with skipping states shown in dashed arcs.

Initially, a constant number of states are used in our models: while using the letter model, a constant number of states is used for each letter and similarly if word models are trained, a constant number of states is used for every word. For observation probabilities, continuous Gaussian probability densities are used. We have tried both a single Gaussian and a mixture of Gaussians. Unless otherwise specified, we used 20 states per letter model with a single Gaussian, without skipping states, as the default setting. The results of the experiments with different HMM settings are explained in Section 3.

The Turkish alphabet contains the letters: a b c **ç** d e f g **ğ** h i j k l m n o **ö** p r s **ş** t u **ü** v y z. Some of the letters with a cedilla or accent, are represented by two unit models. For instance letter ç is modeled as a c followed by '. All remaining letters are represented by single letter models. Representing the similar written parts of different written letters like s and ş aims to make better use of the available training data.

Figure 2 shows a sample word *network* that has word models in parallel. The most probable path taken in the network gives us the most probable word.

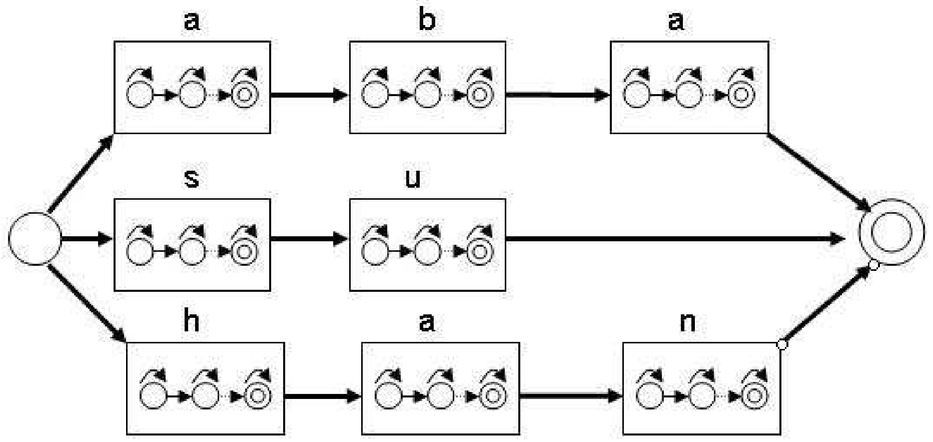

Figure 2. A word network where each word model ("aba" "su", "han") is built up of its constituent letter models.

2.3. Feature Set

In this work, five main features are used: first and second derivatives of the x and y coordinates and percentage change in the pressure. We do not use the value of x and y coordinates as features since they are not translation invariant. For robustness against jitter, the first and second derivative of the coordinates are calculated using equations (3) and (4) respectively:

$$dx_t = \frac{\sum_{\theta=1}^{\Theta} \theta \times (x_{t+\theta} - x_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2}, \tag{3}$$

$$ddx_t = \frac{(dx_{t+1} - dx_{t-1})}{2\Theta}, \tag{4}$$

where x_t is the value of the x-coordinate at time t and Θ, is half the observation window width. The derivative of y-coordinate is found similarly. We used $\Theta = 5$ for this study. Unless otherwise specified, these features are calculated considering the strokes in a word as connected. We experimented with using the stroke information (not considering strokes as connected), as explained in Section 3.2.

The percentage change in pressure is calculated using equation (5) where likewise p_t denotes the pressure value at time t:

$$dp_t = \frac{(p_{t+1} - p_{t-1})}{2p_t}. \tag{5}$$

3. EXPERIMENTS AND RESULTS

3.1. Database

System is developed and tested in two stages. In the first stage, a vocabulary consisting of 50 different words has been determined and the handwriting samples of these words have been collected from 20 people, forming the first database of 1000 words (50 words x 20 people). The interface used for data collection is similar to that of the Tablet PC handwriting applications where users are expected to write on a baseline.

A larger vocabulary set consisting of 1000 words is compiled from e-mail messages, so as to represent the daily vocabulary (of students). This 1000 word vocabulary, is separated into 10 sets, each containing 100 words. Thirty people were recruited for writing one of the sets. Consequently, for each word set, handwriting data from 3 different people is collected, which resulted in a database of size 3000 handwritten words (10 sets x 100 words x 3 people).

Data Set 1 (100 words) 3 people
Data Set 2 (100 words) 3 people
Data Set 3 (100 words) 3 people
● ● ●
Data Set 10 (100 words) 3 people

Figure 3. 10 different data sets totaling 1000 words form the second database.

Both vocabulary sets were designed to have an equal distribution of Turkish characters and the words were selected from most frequently used Turkish words. Each word appears only once in the sets to avoid bias during training and testing.

3.2. Experiments

We have studied the effects of several attributes, such as the HMM topology and different features, in terms of their contribution to the word recognition performance.

Effects of Letter versus Word Models

In this first experiment, the first database is separated into training and test sets: data from 15 people (750 words) is used as the training set and data from the remaining 5 people (250 words) is used as the test set. Hence, the words were common across training and test sets, but the writers were different.

Table 1 shows the success rates of the letter and word model experiments that are carried out using the set mentioned above. Letter model and word models have success performance of 97% and 95% consecutively. Given enough training data, one would expect the word model to give better results. Here, the reason for worse performance may be due to the lack of enough training data and unoptimized number of states for the word model.

Even though we have experimented with word models for this small-vocabulary experiment, as the number of words in the vocabulary increases, using word HMM models becomes infeasible due to the increased number of models that need to be trained (requiring large amounts of data and computation time).

Model	# of Writers	# of Words	# of States	Writer Independent	Correct
Letter	20	50	20	Yes	97 %
Word	20	50	60	Yes	95 %

Table 1. Performance rates for the letter and word models using the first database.

Effects of Number of States on Letter & Word Models

In the second experiment the relationship between the number of states and performance accuracy is studied, keeping the other parameters as in the first experiment. A one Gaussian mixture is used for modeling the observation densities. The recognition results are summarized in Table 2, where the best number of states for the letter model is found to be 20. Optimal number of states for the word models is found to be 70 states on the average (word length average for Turkish is 8-10 letters). Note that the word model performance is improved when more states are used. Nonetheless, as mentioned before, in order to deal with very large lexicons, letter models are necessary; therefore in all of the remaining experiments we use letter models.

Model	**# of Writers**	**# of Words**	**# of States**	**Correct**
Letter	20	50	10	92 %
Letter	20	50	20	97 %
Letter	20	50	30	94 %
Word	20	50	30	89 %
Word	20	50	50	94 %
Word	20	50	70	96 %

Table 2. Impact of number of states in the letter and word model performance using the first database.

Results for the Second Database

After the development of the prototype phase of the project where the above experiments have been conducted, more realistic results have been obtained with the 1000-word sized database. As we have mentioned before, this second database (shown in Figure 3), was separated into 10 sets of 100 words and 30 different people participated in writing one of the sets.

In the third experiment, the second database is split into training and test sets such that they have common words but different writers. Since the second database consists of 10 groups x 100 words x 3 people, for every group two people were used for training and one person was used for testing. The success rate in this experiment is found to be 90.4%, as shown in Table 3. This experiment gives us a baseline performance for the next few experiments that also use the same training and test sets.

Model	**Training Set**		**Test Set**		**Correct**
	# of Writers	# of Words	# of Writers	# of Words	
Letter	20	1000	10	1000	90.4%

Table 3. The performance for the third experiment where the 2nd database was split so that the words are common whereas the writers are different, between the training and test sets.

Skipping State HMM Topology

In this experiment, missing strokes are handled by trying a new topology for the letter HMM models. For this, we allowed for the possibility of skipping states as shown in Figure 1, to allow for more flexible models. As can be seen in Table 4, the success rate is increased by 1% compared to Experiment 3 where skipping states was not allowed.

Model	**Training Set**		**Test Set**		**Topology**	**Correct**
	# of Writers	# of Words	# of Writers	# of Words		
Letter	20	1000	10	1000	non-skipping	90.4%
Letter	20	1000	10	1000	next state skip allowed	91.4%

Table 4. The performance of HMMs with skipping state topology.

Using Different Feature Sets

Since we have limited training data, it makes sense to reduce the dimensionality of the features. In order to study the effects of different feature sets on the performance of the system, we measured the system performance using different subsets of the features described in Section 2.3: the first derivative of the x coordinate (dx), the first derivative of the y coordinate (dy), the second derivative of the x coordinate (ddx), the second derivative of the y coordinate (ddy), and the percentage change in pressure (pressure) between consecutive points.

As can be seen in Table 5, the best performance was obtained using the full set of features, as in Experiment 3. In other words feature reduction was not found to be useful, in this case. Therefore, in all of the remaining experiments, we use all of the five features (dx, dy, ddx, ddy, pressure).

Model	**Training Set**		**Test Set**		**Features**	**Correct**
	# of Writers	# of Words	# of Writers	# of Words		
Letter	20	1000	10	1000	dx,dy,ddx,ddy,pressure	90.4%
Letter	20	1000	10	1000	dx,dy	89.3%
Letter	20	1000	10	1000	dx,dy,ddx,ddy	88.7%

Table 5. Performance rates for different feature sets.

Using a Mixture Model

Table 6 shows the effects of using a Gaussian mixture, increasing the number of Gaussians and varying the number of states accordingly. Given enough training data and sufficient number of states, one would expect increased number of Gaussians to better model the observation probabilities, however for this small experiment, we did not obtain a significant performance increase. Using a three-Gaussian mixture and 7 states we had 90.5% performance compared to the baseline performance of 90.4% where a single Gaussian was used (default setting). Since this is a very small performance increase, we did not use multiple Gaussians in the following experiments.

Model	**Training Set**		**Test Set**		**# Of States**	**# Of Mixtures**	**Correct**
	# of Writers	# of Words	# of Writers	# of Words			
Letter	20	1000	10	1000	20	1	90.4%
Letter	20	1000	10	1000	7	3	90.5%
Letter	20	1000	10	1000	10	3	89.7%

Table 6. The performance of different mixture and state numbers.

Variable Number of States for Different Letters

In this experiment, we grouped the letters roughly according to their shape, complexity and number of strokes into seven groups and varied the number of states used in the corresponding HMM models between 7 and 25, such that simpler groups had smaller number of states.

Compared to the baseline performance of Experiment 3 where 20 states were used for each letter model, the use of different number of states for letters did not bring any significant improvement (increase from 90.4% to 90.5%). Even though one would expect to obtain better results with this approach, the clustering of the letters and assignment of states was done in an ad hoc manner; grouping the letters into 2 or 3 groups and using a smaller range for the number of states (around 20) may be useful. For the remaining experiments, we kept the number of states a constant (20) across different letter models.

Different Partitioning of the DataBase

For this experiment, we split the second database into training and testing as follows: two sets (total of 200 words) have been used for testing and 8 sets (total of 800 words) have been used for training. The goal was to increase the number of training data (from 66% from the previous experiments to 80%). Note that this way, *both* the words and the writers differ between the training and the tests sets, since the sets of 100-words are mutually exclusive.

In order to measure the performance robustly, the database is split into training and test sets in 5 different ways and the average performance is computed (e.g. Data Sets 1 and 2 are used for testing, and the rest are used for training). Table 7 shows the results for different distributions: one can see that variations across different splits is small, with an average performance of 91.1%.

As we expected, the increased training data gave a better performance compared to the baseline performance (90.4%) of Experiment 3, even though words in the test are not included in the training set. Note that word models are not convenient for this experiment, since using word models, the domain of test words should be a subset of train words.

Model	**Training Set**			**Test Set**			**Correct**
	# of Writers	# of Words	Data Sets	# of Writers	# of Words	Data Sets	
Letter	24	800	DS3..DS10	6	200	DS1,DS2	92.6%
Letter	24	800	DS1,DS2,DS5..DS10	6	200	DS3,DS4	92.5%
Letter	24	800	DS1..DS4,DS7..DS10	6	200	DS5,DS6	89.6%
Letter	24	800	DS1..DS6,DS9,DS10	6	200	DS7,DS8	89.8%
Letter	24	800	DS1..DS8	6	200	DS9,DS10	91.2%
						Average:	**91.1%**

Table 7. Different test set performance rates using letter models and the second database. Both writers and words are different between the training and test sets.

Extracting Features Within a Stroke

In online handwriting recognition, letters written in multiple strokes increase the complexity of the problem, due to the transitions between the strokes. In our case, we have also noticed that the features used (dx,dy,ddx,ddy) vary significantly according to whether a word is written cursively or discretely, or whether the second stroke was done without lifting the pen.

If the first and second derivatives of the stroke starts are initialized to zero, features are calculated within a stroke. When the stroke transition points are not considered, the adjacent strokes have no effect in stroke feature values, decreasing the variability across different realizations of the same letter, especially in discrete handwriting. As a result, the complexity and variability coming from different pre-strokes and post-strokes are eliminated.

A disadvantage of this approach occurs in letters written with multiple strokes, such as i, ç, ğ, ş, t, k. These letters lose the relationship (relative location etc.) between their strokes and consecutively the recognition rate for these letters might decrease.

Overall, the performance of the system with strokewise feature extraction is 91.8%, showing a 2% increase in performance. Table 8 shows the result compared to the previous experiment with the same train and test sets.

Model	**Training Sets**	**Test Sets**	**Strokewise**	**Correct**
Letter	DS1..DS6,DS9,DS10	DS7,DS8	No	89.8%
Letter	DS1..DS6,DS9,DS10	DS7,DS8	Yes	91.8%

Table 8. The performance of within stroke feature extraction technique, compared to its baseline.

Previous Neighboring Points as a New Feature

In the error analysis we have noticed that the pairs "u" and "a", "y" and "g", "b" and "k", and other similar character pairs were frequently confused by the system. To overcome this problem, we added another feature, computed at each point along the word: number of neighboring points. That is the number of previous points along the stroke which are close (within a fixed distance) to the current point are counted and used, in addition to dx, dy, ddx, ddy, and pressure features.

Using previous neighboring points as a new feature we had 91.3% performance compared to the same experiment without this feature (with performance of 89.8%), as shown in Table 9. Overall performance of the system is increased about 1.5%. Some words that are predicted correctly by this new feature and were previously not recognized are shown in Figure 4.

Model	Training Sets	Test Sets	Previous Pts.	Correct
Letter	DS1..DS6,DS9,DS10	DS7,DS8	No	89.8%
Letter	DS1..DS6,DS9,DS10	DS7,DS8	Yes	91.3%

Table 9. The effect of the previous neighboring points feature, compared to its baseline. Performance of the system is increased by 1.5%.

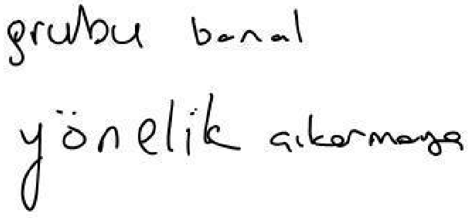

Figure 4. The word “grubu” was recognized as “gruba” and the word “banal” was predicted as “kabul” prior to adding the previous neighboring points-feature. Similarly the word “yönelik” was recognized as “gelip” and the word “çıkarmaya” was recognized as “yıkamaya”. These words are correctly recognized with the new feature.

Relative Height as a New Feature

Another noticeable error in the system was that some letter pairs, such as “b” and “p” , are imperceptible to the system if they are written in two strokes, without knowing the relative height information. Similarly the characters “a” and “d” are difficult to differentiate if the hook of “d” is not discernible.

This problem is partly solved by the addition of the previous feature, but another useful feature would be the relative height of each point with respect to the baseline. Hence the top points on the ascender of the “b” would have a large relative height compared to the top points of the descender of “p”. In order to compute this feature, the baseline, topline and ascender and descender lines are extracted by the system.

Some new errors can occur because of this feature since it is impossible to extract these lines without knowing the underlying word, in some instances. For example when writing the word “şu” one can take below the cedilla as the baseline or below the letter “u”. Although there can be some complications, most of the time the correct ascender and descender lines are extracted and the overall performance is increased, as shown in Table 10. Figure 5 shows some words that were not correctly recognized before, which now are recognized.

Model	Training Sets	Test Sets	Relative Height	Correct
Letter	DS1..DS6,DS9,DS10	DS7,DS8	No	89.8%
Letter	DS1..DS6,DS9,DS10	DS7,DS8	Yes	92.5%

Table 10. The performance of relative height feature, compared to its baseline. Performance of the system is increased by about 2.5%

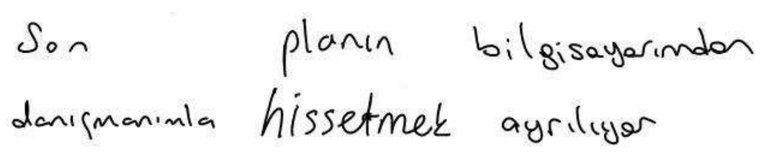

Figure 5. The word “Son” was recognized as “son” and the word “planın” was predicted as “bunun” prior to adding previous intersecting points features Similarly, the words “bilgisayarımdan”, “danışmanımla”, “hissetmek”, “ayrılıyor” were predicted as “bilgisayarıma”, “adımlarla”, “hissettiğimiz”, “duruyor” respectively. These words are correctly recognized with the new feature.

Combining Features

Table 11 shows the results after combining the new features in one test. For this experiment, we used the previous neighbors and the relative height along with strokewise feature extraction. The performance of these features add up and overall performance is increased by 3%. The average performance is 94.0% compared to experiment three (91.1%).

Test Set	Performance (5 features)	Performance (7 features + strokewise extraction)
DS1,DS2	92.6%	95.2%
DS3,DS4	92.5%	95.4%
DS5,DS6	89.6%	91.7%
DS7,DS8	89.8%	94.3%
DS9,DS10	91.2%	93.4%
Average:	**91.1%**	**94.0%**

Table 11. Effects of combining various features with respect to the baseline system.

Combining the New Features and the Skipping State Topology

We have also combined the new features and the skipping state topology and measured the performance on the baseline experiment on data sets DS7 and DS8, with a performance of 89.8%.

The result of this experiment is 94.8% which is a 5% increase from the baseline. However as can be seen in Table 11, the new features alone gave a performance of 94.3%, so the skipping state topology only gave 0.5% additional increase.

4. RESULTS AND CONCLUSION

We have presented an online handwriting recognition system for Turkish. The results are quite good and show the promise of the features and the overall approach of the system.

Sample handwritings from the database are shown in Figure 6. Most words were written discretely (non-cursive), with occasional touching characters, which, unlike offline handwriting, do not pose extra difficulty for the recognizer. However, on close inspection of the errors made by the system, we have seen that while writing a word, many people have gone back and rewritten a bad character somewhere in the word. Since our word models are composed of concatenated letter models of its constituent letters, *in order*, this writing behavior may account for a significant part of the error rate.

We have discovered that more than 35% of the errors are due to the mixed order of strokes or letters, not modeled by the current system (indicated in Table 4 as accented letters and delayed strokes). Mixed stroke order in accented Turkish letters ç, ğ, ş,ü, ö account for 22.7% of the total error. These accents or cedillas are written either right after the main letter body or at the end of the word, or something in between. Stroke ordering is a problem in online word recognition in general, but it is more so in Turkish where many of the characters of the alphabet have dots or cedillas. For further research in letter model the delayed strokes are planned to be handled with a separate model which can be used to extend the word models to allow for alternative orders. Another

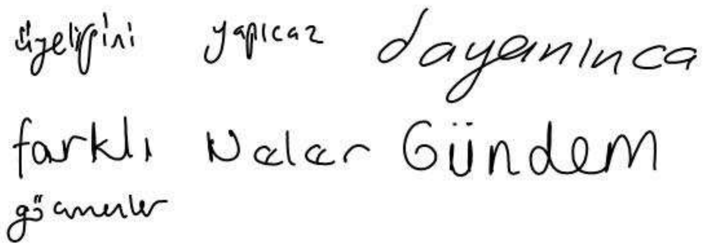

Figure 6. Sample handwritings from the database.

Error Type	Percentage
Accented Letters	22.7%
Delayed Strokes	13.6%
Similar Letters	31.8%
Wrong Baseline, Topline Extraction	13.6%
Miswriting	4.5%
Not Writing Ascenders or Descenders Very Long	13.6%

Table 12. Error Types

form of mixed order is caused by writers correcting a word, after writing the last letter (delayed strokes). This accounts for 13.6% of the total error. As mentioned before, the current word models do not handle this situation, however one can preprocess the word to detect and reorder the letters to some extent, prior to recognition.

Similar to speech recognition systems the system's performance is inversely proportional with the word count so as the vocabulary increases the system's performance as expected decreases. But this 1000 word dataset is compiled from e-mail messages and already covers the different extensions of the same root word. (example: bize, bizler). For the above reason, the system's performance is not expected to decrease significantly as the number of words increase. This prototype system consisting of 1000-word vocabulary will be integrated with the language models and language processing tools for an unlimited vocabulary system, as in the offline handwriting recognition system.[9]

REFERENCES

1. P. Artieres, "Stroke level HMM for on-line handwriting recognition," *Proc. of IWFHR-8*, 2002.
2. D. Li, A. Biem, and J. Subrahmonia, "Hmm topology optimization for handwriting recognition," *Proc. of IEEE* **3**, pp. 1521–1524, 2001.
3. S. Connell and A. K. Jain, "Template-based online character recognition," *Pattern Recognition* **Vol. 34 (1)**, pp. 1–14, 2001.
4. J. Hu, S. G. Lim, and M. K. Brown, "Writer independent on-line handwriting recognition using an HMM approach," *Pattern Recognition* , 2000.
5. F. Wang, L. Vuurpijl, and L. Schomaker, "Support vector machines for the classification of Western handwritten capitals," *Proc. of IWFHR-7*, pp. 167–176, 2000.
6. R. Plamondon, D. P. Lopresti, L. R. B. Schomaker, and R. Srihari, "On-line handwriting recognition," in *Wiley Encyclopedia of Electrical & Electronics Engineering*, J. Webster, ed., pp. 123–146, 1999.
7. E. J. Bellegarda, J. R. Bellegarda, D. Nahamoo, and K. S. Nathan, "A fast statistical mixture algorithm for on-line handwriting recognition," *IEEE Trans. PAMI* **16**, pp. 1227 – 1233, 1994.
8. "http://msdn.microsoft.com/msdnmag/issues/03/10/ tabletpc/default.aspx,"
9. B. Yanikoglu and A. Kholmatov, "Turkish handwritten text recognition: A case of agglutinative languages," *Proc. SPIE*, 2003.
10. K. Oflazer, "Two-level description of Turkish morphology," *Literary and Linguistic Computing* **9(2)**, p. 1994.
11. S. Young, *The HTK Book v3.0.*, Cambridge University, 1999.